Universidad
Carlos III de Madrid
www.uc3m.es

BACHELOR THESIS

Title: Creation of an English Ontology for Oncological Environments Applied to Chatbox

Author: Ionut Sorin Rusu

Degree: Computer Science Degree

Teacher: Anabel Fraga Vázquez

Date: March, 2015

Table of Contents

1. Introduction

Information technologies have heavily influenced our lives the last decades, helping and advancing – and sometimes redefining and revolutionizing – areas like education, medicine, sports, communications, entertainment, and many others. One of the most important of the aforementioned subjects is that of medicine: a great effort (measured in time and money) has been put to apply IT for improving people's health individually or institutionally, that is, by means of technologies of particular use by the patient or technologies used in hospitals, thus directly affecting not only patients, but doctors too. It is in this area, then, on which this project is focused in: IT applied to hospitals for the individual's benefit, or, more specifically, software making available for any English-speaking Internet user information related to Spanish hospitals, focused on the oncology field of medicine.

1.1. Scope and Goals

The aim of the project was to extend an already existing platform, the Semantic Virtual Assistant developed by the Knowledge Reuse research group of Carlos III University, so as to be able to answer user questions related to hospitals (location, phone number, type of hospital, doctors, oncology specialization) in English. The project is part of a coordinated project in collaboration with the University of Granada, belonging to the National Plan.

The included hospitals are those located on the Spanish territory, and the existing doctors in the system are those specialized in oncology, without trying to provide an exhaustive list.

This present paper aims to document the implementation and results of the aforementioned new features built into the Semantic Virtual Assistant, as well as providing its design and architectural details and also adding contextual information related to the topics involved in the project: Virtual Assistants and IT solutions in medical systems.

Firstly, the state of the art of Virtual Assistants, Medical Aid Systems and of the used technologies will be presented, also providing historical information about its development. Secondly, the analysis of the project will be set forth, which includes the requirements, the architectural design and the detailed design. Thirdly, the implementation and testing of the solution will be explained. Then the conclusions and further improvements will be expounded, which is followed by information related to the planning and the budget of the project. Lastly, the used references and bibliography will be provided.

1.2. Acronyms

Acronym	Identifier
VA	Virtual Assistant
SVA	Semantic Virtual Assistant
VS	Visual Studio
VB	Visual Basic
ASP	Active Server Pages
JSP	Java Server Pages
PHP	Personal Home Page
IIS	Internet Information Service
OS	Operating System
MDB	Microsoft Access Database
PDF	Portable Document Format
EDR	Enterprise Domain Reuser
KR	Knowledge Reuse

2. State of the art

In this section three topics will be treated: Virtual Assistants, Medical Aid Systems, and the employed technologies in the implementation of the Virtual Assistant, which are: Microsoft Visual Studio, Microsoft .NET, Microsoft SQL Server, and Microsoft IIS Services. Information related to their history, evolution and relevance will be provided

2.1. Virtual Assistants

A virtual assistant (VA) can refer to two things, at the highest level: to a real person that provides services to other people without co-location, but at a distance through the Internet or through telephone; and to a virtual person, that is, software explicitly programmed to meet the user needs, also called Intelligent Virtual Assistant (IVA) or Intelligent Assistant (IA)[i]. The second kind of VA will be treated in this report.

At the same time, the software-based VAs can be separated in two broad categories: generic and specialized. Generic Virtual Assistants are those created for helping the user in a variety of contexts, not focused on a single area of knowledge. In this category we can find, for example, Apple's *Siri*, Microsoft's *Cortana*, Google's *Google Now* and IBM's *Watson* They also resemble more faithfully what a traditional assistant is, since they are agents that can perform tasks on the user's account. Specialized Virtual Assistants, on the contrary, are only operative on one main area of knowledge, a domain, and they will not perform actions such as buying or scheduling an appointment at the user's command, but instead will provide information. Examples of this kind of Virtual Assistants are Nuance's *Nina*, Renfe's *Irene* and Ikea's *Anna*. Although both kinds share a very important basis, the Natural Language Processing, they emerged from at different times, have a different background, a different purpose, and use different technologies. Since the area of work of this thesis is an information-providing VA, Generic Virtual Assistants will not be treated.

2.1.1. A History of Virtual Assistants

If a Virtual Assistant is defined as software that interacts with humans in natural language, the technology is not so recent. First were born the *chatbots* (or *chatterbots*), that simulated a conversation without having any area of knowledge and therefore not used for providing the user with information. Their purpose can be understood better if the Turing Test is acknowledged: it was proposed by Alan Turing in his 1950 article "Computing Machinery and Intelligence" and it can be said to consist in two actors, a judge (A) and a subject (B). (B), sometimes a machine and sometimes a human, would have to fool (A) into thinking that he is a human when being a machine. Basically, (A) has to distinguish when (B) was a machine and when a human. If a machine was able to drive the judge into thinking it was a human, then it had passed the Turing Test. The definition of the test provided by Turing is not clear, as it happens with its purpose.[ii]

The *chatterbots* were then designed with Turing's Test in mind: their purpose is not to solve a user's problems or to provide him with information, but to maintain a coherent conversation, to simulate intelligent conversation. Among the first and most famous *chatbots* we find ELIZA, emulating a Rogerian Psychotherapist, developed by Weizenbaum from the MIT in 1966. Weizenbaum himself plugged ELIZA off when secretaries and nontechnical staff thought that it

was a real therapist, spending hours talking to it about their personal problems. This terrified him and led him to be an active oppose to this kind of development.[iii] His critiques effectively led to a slowing down in the research and development of similar projects.

In 1972 psychiatrist Kenneth Colby programmed PARRY, a *chatbot* that simulated a paranoid schizophrenic. Only 48% of the psychiatrists were able to identify PARRY as a computer program, therefore passing the Turing Test. Given that Turing's Test rules are not clear, that the percentage of correct identifications is so close to random guessing, and that many psychiatrists did not really know the capabilities of a computer, the affirmation that PARRY passed the Turing Test is not widely accepted.

Chatbots haven't evolved too much since then, and ELIZA and PARRY are among the most interesting ones. Alice (A.L.I.C.E, Artificial Linguistic Internet Computer Entity), programmed in 1995 and inspired by ELIZA, is one of the best in its category and was awarded the Loebner Prize (the oldest Turing Test contest) three times.[iv] Rose, a San Francisco 20-something girl who works as a hacker and is a thorough skeptic, created by Bruce Wilcox, won the last Loebner Prize in November 2014.[v]

Chatbots, as stated, are usually not useful by themselves; they are not assistants, or helpers. However, going from a *chatbots* to a useful Virtual Assistants that can be employed on a wide range of businesses is not a costly step. The architecture, the interface and the systems involved in it are the same: it's only the area of knowledge of the *chatterbot* that has to be enriched.

Building on top of the traditional *chatbots*, it was IKEA, in 2003, that made a Virtual Assistant available to the larger public. Still now, Anna (the VA) is one of the most known *chatbots*. She is present in more than 20 countries and is able to understand 21 different languages, answering questions about products, prices, opening hours, etc.

Since the advent of Anna, many other businesses have decided to implement VAs due to the high savings they are able to make in the customer communication departments. What's more, they usually improve customer experience and produce better customer satisfaction in that there are no waiting times and in that they are just one click away. These businesses range from banks (Commercial Bank of Dubai's Sarah) to marketing agencies (Your Online Marketing Agency's Toby) and public services (Anglian Water Services' Amanda).

2.1.2. The Future of Virtual Assistants

The benefits obtained out of Virtual Assistants attract increasingly more businesses. The adoption of VAs is very similar to what happened with web pages: it became a necessity for many businesses to have a web page. Although the extent of VAs may never be as wide as that of web pages, it certainly follows the same trend – maybe at a slower pace. It can be stated that, without any doubt, VAs have not reached their full extent, and users will be able to benefit their services in many other websites over the next years.

VAs are advancing not only in extension: the technologies that lie behind are also improving. Efforts are directed mostly to the user interface, as is the case of Zoe, the Virtual Assistant developed by researchers at the University of Cambridge. Zoe is able to show emotions, in her

facial expression as well as in her voice. Efforts are also put in obtaining better speech recognition systems so that the Human-Computer Interaction would be more natural.

2.2. Medical Aid Systems

Technology is positively affecting many industries, if not all. Among them is the healthcare industry, one of the most growing (in concern and investment) industries of the last decades. More recently, a shift is taking place in how the healthcare system works: traditionally it consisted in business to business operations, but nowadays the individual has more options and power over his own health and the services provided to him.[vi] The outcome of this shift relating to technology is that its impact is no longer only on developing better machines that can more effectively and efficiently cure illnesses; technology is also applied to offer the patient a more customized experience. The industry is becoming more customer-centered, and so are the technologies involved.

Richard Satava, of the University of Washington, stated: "The future of technology – and medicine in general – is not in blood and guts but in bits and bytes."[vii] He suggests that all the techniques and methods used in hospitals will and should involve computers: storing patients' data, see the patients' body status on a screen, interact with them and doing surgical operations remotely, etc.

One example of how technology impacts healthcare is a telemedicine solution, RP-VITA, the first remote presence robot. The robot can be used in three different environments: as a substitute for the doctor or nurse, being able to travel autonomously from one room to another; as a help to doctors, providing and storing data about the patient; and, lastly, as the medium to let relatives be involved in decisions remotely. It is equipped with two screens (one for display and a touchscreen), a camera prepared for facial recognition or for reading RFIDs and speech recognition to understand orders, among other features. An app exists for Apple's iPad that allows the doctors control the robot from anywhere.

Less institutionary than the RP-VITA, but with a greater impact, is the Internet itself. There are especially-created websites that provide a diagnosis for anyone with access to the Internet. One example is YourDiagnosis: besides making suggestions about possible diseases based on symptoms, it also stores that information along with the whole health record and makes it easy to share with health professionals. Microsoft's HealthVault has similar functions.

Nuance's involvement in healthcare is worth mentioning. Nuance, focused on speech recognition (having also created Nina, one of the most intelligent VAs), has created many products related to clinical documentation. Doctors may more easily create reports or skip the note taking process while reviewing a patient by using their speech recognition software able to transcribe conversations.

Technology affects every aspect of healthcare, and it is only the beginning. Healthcare being such a big industry, the efforts of many IT companies are and will be directed towards improving what already exists, and innovate, creating what doesn't.

2.3. Employed Technologies

In this section the technologies used in the development of the Virtual Assistant will be discussed: Microsoft Visual Studio 2008, Visual Basic for .NET (VB.NET), ASP.NET, Microsoft SQL Server 2005, Microsoft SQL Server Management Studio 2008 and Internet Information Services (IIS).

2.3.1. Microsoft Visual Studio 2008

The Knowledge Reuse group used Visual Studio for developing the Virtual Assistant (as well as all the other related tools that are part of the VA ecosystem) in its entirety: from the business to the presentation layer.

Visual Studio is an IDE (Integrated Development Environment) created by Microsoft to help developers program applications for Windows and .NET platforms in an easier and more effective way. The first edition of Visual Studio appeared in 1997, although Microsoft distributed development tools for specific programming languages before (e.g. for VBA and C++). The IDE is one of the most valued by developers, one of the factors being IntelliSense, an auto-completion tool that works smoothly and gives great results.

VS also supports developing in ASP (Active Server Pages), empowering the user to build fully functional web applications using only one IDE.

Microsoft distributes Visual Studio in different editions. The Express edition is completely free, while the Professional and Ultimate (known as Team Edition before VS 2010) editions, which include features like advanced testing, are paid for.

Visual Studio 2008 is the sixth version of Visual Studio, released on November 19, 2007. It has in-built support for Visual C++, Visual Basic and Visual C#. Visual F# and JScript were not available in the 2008 version, but integrated in later versions. Other programming languages are supported through extensions.

2.3.2. Visual Basic for .NET

Among the programming languages supported by Visual Studio one of the earlier and most important is Visual Basic. Developed by Microsoft, Visual Basic was first released in 1991. Created especially for rapid application development, it went through significant improvements until its most popular version, Visual Basic 6.0, released in 1998 and having its support period ended by Microsoft in 2008[viii].

The next versions of Visual Basic are built into the .NET platform, aiming at creating .NET, Windows or Web applications and Web Services. A new version of Visual Basic.NET is released with each new .NET upgrade, coinciding with new Visual Studio versions. In 2015 VB.NET 14 is planned to be released, version 7.0 having been released in 2002, 7.1 in 2003, 8.0 in 2005, 9.0 in 2008, 10.0 in 2010 and 11.0 in 2012.

The business layer and the web services were built using VB.NET 8.0 in .NET version 2.0.

2.3.3. ASP.NET

ASP.NET is a web application framework developed by Microsoft as part of the .NET platform. It allows the creation of dynamic websites, web applications and web services. The first version

was released in 2002, and it came to substitute ASP (similar to VB and VB.NET).

The functionality that ASP.NET offers is similar to that of PHP or JSP: it allows the insertion of code running on server side in the presentation files (suffixed as .aspx), although VB.NET encourages placing the dynamic code outside the .aspx files, contributing to a separation between the presentation and the business layer.

ASP.NET applications work in conjunction with an IIS server.

2.3.4. Internet Information Services

The Virtual Assistant is made available through the internet by using IIS, Microsoft's web server. As a webserver, it processes http requests from the users, and answers in the form of webpages.

Since the advent of IIS 1.0, new versions were released almost parallel to new Windows Operating Systems. Windows 8.1, the latest Windows OS, runs IIS 8.5. The Virtual Assistant, residing in a Windows 7 OS, uses IIS 7.5.

2.3.5. Microsoft SQL Server 2005

SQL Server is Microsoft's database management system. It allows the user and other applications to store and query data in relational databases mainly through Transact-SQL query language.

SQL Server was born in the late 1980s as the result of the collaboration between Microsoft and Sybase Corporation, running on IBM's OS/2 operating system. The collaboration between the mentioned companies ended with SQL Server 6.0, when Microsoft decided to ship the product exclusively for its NT operating system, although a previous version was compatible with NT. In further versions the software would belong entirely to Microsoft.[ix]

SQL Server 2005, released on October of the same year, is the fifth version of SQL Server (if SQL Server 6.0 is considered to be the first release, as it usually is). There exist three other major SQL Server versions: SQL Server 2008, SQL Server 2012 and SQL Server 2014.

2.3.6. Microsoft SQL Server Management Studio 2008

SQL Server Management Studio is an application that allows managing the whole range of features of SQL Server. It gives the user a visual way of configuring and managing the database system, also allowing him to create scripts. Creating a new database, new tables, relationships, querying, editing records… it all can be done in just a few clicks.

3. Analysis

In this section the project's technical requirements will be listed, and the Virtual Assistant's architectural and detailed design will be provided.

3.1. Requirements

3.1.1. User Requirements

Requirement ID	URQ_1	Name	English Translation		
Necessity	High	Priority	High	Dependency	
Description	The Virtual Assistant shall understand English and answer accordingly.				

Requirement ID	URQ_2	Name	Hospitals Information		
Necessity	High	Priority	High	Dependency	URQ_1
Description	The Virtual Assistant shall provide the following information about Spanish hospitals: name, address, type and phone number.				

Requirement ID	URQ_3	Name	Doctors Information		
Necessity	Medium	Priority	Medium	Dependency	URQ_1
Description	The Virtual Assistant shall provide information about the required doctors.				

3.1.2. Design Requirements

Requirement ID	RQ_1	Name	Database Migration		
Necessity	High	Priority	High	Dependency	
Description	The content of the tables Answers_Special_Terms, Answers, Artifact_ Answer, Structured_Questions, Special_Terms and Tag_Patterns contained in the database VirtualAssistant(Spanish) shall be migrated to the database VirtualAssistant(English).				

Requirement ID	RQ_2	Name	Database Replication		
Necessity	High	Priority	High	Dependency	
Description	The database VirtualAssistantData(Spanish) shall be replicated as VirtualAssistantData(English).				

Requirement ID	RQ_3	Name	English Translation		
Necessity	High	Priority	High	Dependency	RQ_1
Description	The tables Answers, Structured_Questions, Special_Terms and Tag_Patterns from the database VirtualAssistant(English) shall be translated into English.				

Requirement ID	RQ_4	Name	English Translation of Data		
Necessity	High	Priority	High	Dependency	RQ_2
Description	The tables Adjetivos, Capitales, Colores, Descubrimientos, Dias, Gustos, Lugar, Pares, Personajes, Preferencias, Productos, QueEs, Servicios and Tiempo from the database VirtualAssistantData(English) shall be translated into English.				

Requirement ID	RQ_5	Name	Hospital Type Table		
Necessity	High	Priority	High	Dependency	RQ_2
Description	The table HospitalType with the fields Id_HospitalType and HospitalType shall be created in the database VirtualAssistantData(English).				

Requirement ID	RQ_6	Name	Hospital Table		
Necessity	High	Priority	High	Dependency	RQ_2, RQ_5
Description	The table Hospitals with the fields Id_Hospital, Name, Address, PostaCode, AutCommunity, Province, City, Telephone, Type and Organization shall be created in the database VirtualAssistantData(English) with the field Type as a foreign key pointing to the field Id_HospitalType from the table HospitalTye.				

Requirement ID	RQ_7	Name	Specialty Table		
Necessity	High	Priority	High	Dependency	RQ_2
Description	The table DoctorSpecialty with the fields Id_DoctorSpecialty and DoctorSpecialty shall be created in the database VirtualAssistantData(English).				

Requirement ID	RQ_8	Name	Doctors Table		
Necessity	High	Priority	High	Dependency	RQ_2, RQ_7, RQ_6
Description	The table Doctors with the fields Id_Medico, Id_Hospital, Name, Specialty, PhoneNumber and Email shall be created in the database VirtualAssistantData(English) with the fields Id_Hospital pointing to the field Id_Hospital from the table Hospitals and Specialty pointing to Id_DoctorSpecialty from the table DoctorSpecialty as foreign keys.				

Requirement ID	RQ_9	Name	Hospital Type Population		
Necessity	High	Priority	High	Dependency	RQ_2, RQ_5
Description	The table HospitalType shall be populated.				

Requirement ID	RQ_10	Name	Hospitals Population		
Necessity	High	Priority	High	Dependency	RQ_2, RQ_5, RQ_6
Description	The table Hospitals shall be populated.				

Requirement ID	RQ_11	Name	Doctor Specialty Population		
Necessity	High	Priority	High	Dependency	RQ_2, RQ_8
Description	The table DoctorSpecialty shall be populated.				

Requirement ID	RQ_12	Name	Doctors Population		
Necessity	High	Priority	High	Dependency	RQ_2, RQ_8, RQ_11
Description	The table Doctors shall be populated.				

Requirement ID	RQ_13	Name	Virtual Assistant Service		
Necessity	High	Priority	High	Dependency	
Description	A new Virtual Assistant Service that connects to VirtualAssistant(English) database shall be created using the Virtual Assistant Service Manager application.				

Requirement ID	RQ_14	Name	Clarifier Rules		
Necessity	High	Priority	High	Dependency	
Description	The following Clarifier Rules shall be created: • When there is a noun followed by the word *address*, *address* will be considered to be a noun, not a verb. • When there is a noun followed by the word *phone*, *phone* will be considered to be a noun, not a verb. • When there is a noun followed by the word *email*, *email* will be considered to be a noun, not a verb. • *What*, when followed by a modal verb, will be considered to be an utterance determiner, not a relative pronoun. • When a pronoun is preceded by a phrasal verb base, the following element will be taken as a phrasal verb article, not as a preposition.				

Requirement ID	RQ_15	Name	Phrase Patterns		
Necessity	High	Priority	High	Dependency	
Description	Phrase Patterns able to recognize the following question patterns shall be created: • What is <hospital name>'s location? • Where is <hospital name>? • What's <hospital name>'s address? • What oncology doctors are there at <hospital name>? • What's <hospital name>'s phone number? • What's <doctor name>'s phone number? • What's <doctor name>'s email address? • How can I contact <doctor name>? • Who is <doctor name>? • What can you tell me about <doctor name>? • What do you know about <doctor name>? • What hospitals in <location> have an oncology department? • What oncology centers are there in <location>? • Where can I find a <hospital type> hospital? • Can you list me <number> oncology centers?				

Requirement ID	RQ_16	Name	SQL statements		
Necessity	High	Priority	High	Dependency	RQ_15
Description	SQL statements shall be created to extract the information corresponding for each Phrase Pattern.				

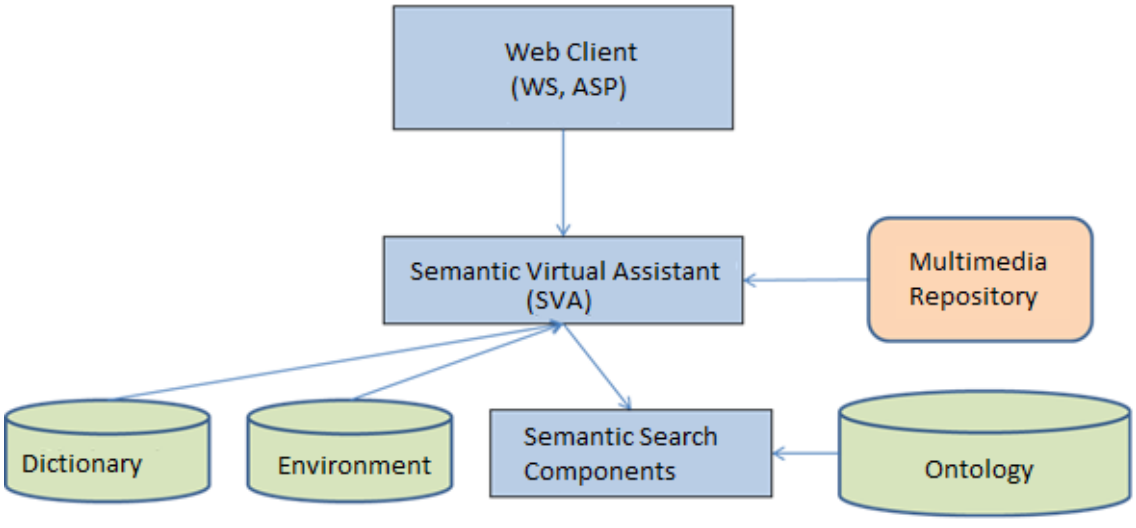
Requirement ID	RQ_17	Name	Database Phrase Patterns		
Necessity	High	Priority	High	Dependency	RQ_15, RQ_16
Description	Rows containing the fields ID, pivot, QuestionType, NoResultAnswer, Question, VerbSemantic, NumVariables, Multivalue, SQLStatement DatabaseServer, DatabaseName, DatabaseLogin and DatabasePassword associated to each phrase pattern shall be created in the Structured_Questions table.				

3.2. Architectural Design

The Semantic Virtual Assistant is composed of three layers:

- A web layer that interacts with users: it sends their queries to the semantic server and presents the results to the user.
- A web abstraction layer that makes available on the Internet the semantic service.
- The semantic service that manages every user session in order to obtain the queries' results and generate a coherent dialogue in order to make easier the interaction between the user and the computer.

The Semantic Virtual Assistant´s architectural diagram can be seen in the following illustration:



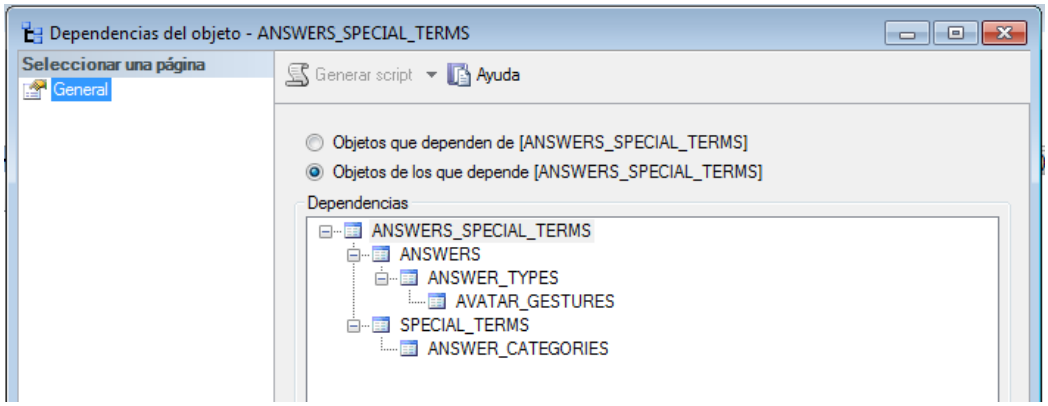
4. Implementation and Testing

The steps, methods and tools used for fulfilling the requirements will be explained in this section, along with the testing results.

4.1. Database Migration

In order to migrate the content of the tables listed in the requirement RQ_1 from the *VirtualAssistant(Spanish)* to *VirtualAssistant(English)* database, Microsoft SQL Server Management Studio 2008 has been used.

Besides constructing the right load and insert queries, the order had to be taken into account, since some tables reference through foreign keys fields from other tables. This is especially critical in intermediate tables. For example, the table *Answers_Special_Terms* is an intermediate table between *Answers* and *Special_Terms* (that also depend on other tables), as shown in the following illustration:



The load and insert queries follow the next structure:

```
INSERT INTO <target database name> [(<field 1>, <field 2>…<field n>)]
SELECT [(<field 1>, <field 2>…<field n>) | *] FROM <origin database
name> [WHERE <condition 1> [,… <condition n>]]
```

In order to be executed, a query window must be opened by clicking the new query button (Nueva consulta), pasting the insert query, and clicking the execute button (Ejecutar). The insert query, however, doesn't work unless the following line of code is written before the query statement:

```
SET IDENTITY_INSERT <target database name> ON
```

This is due to a restriction related to primary key fields of the table.

Since some of the target tables were not empty, their content had to be deleted before the inserts. This is done with the following SQL statement:

```
TRUNCATE TABLE <target database name>
```

Finally, some tables contained values applicable only to the original environment of the SVA, and therefore had to be changed. This was the case of the *DatabaseServer* field of the

Structured_Questions table, which contained the KR group server, and had to be changed to the local server. The field update statements must follow the next syntax:

```
UPDATE <target database name> SET <field name>=<new value>
[WHERE <field name>=<old value>]
```

4.2. Database Replication

As specified in RQ_2, the database *VirtualAssistantData(Spanish)* had to be replicated as *VirtualAssistantData(English)*. In order to do this, a backup must be saved to the file system first. This step is accomplished through the Microsoft SQL Server Management Studio 2008, where the database must be right-clicked and the *Backup Copy* option must be selected in the *Tasks* cascade menu. Not being necessary to modify any other option, after the route for the backup file is specified and the accept button is clicked, a backup copy will be saved into the computer.

The next step is to recover the backed-up database. By right-clicking the *Databases* folder in Microsoft SQL Server Management Studio 2008 and choosing the *Restore Database* option, a new window pops up, where the file to be restored must be selected from the computer's file system and a new name must be given to it (*VirtualAssistantData(English)* in this case).

After these two steps have been accomplished the new database will be available in the *Database* folder.

4.3. Databases Translation

Requirements RQ_3 and RQ_4 list the tables from *VirtualAssistant(English)* and *VirtualAssistantData(English)* that must be translated into English. This is also done in Microsoft SQL Server Management Studio 2008: the desired table must be right-clicked and the *'Edit the first 200 rows'* option must be selected. The result is a new tab with the list of all the table's fields and the first 200 rows, which are modifiable. By switching to the SQL view () the 200 rows limitation can be erased by deleting the `TOP(200)` line of code.

Executing the modified statement a list with all the table rows will be available, and all of its values modifiable. The translation is straightforward.

4.4. New Tables Creation

Requirements RQ_5, RQ_6, RQ_7 and RQ_8 specify each a table to be created in the *VirtualAssistantData(English)* database. The relational representation is the following:

HospitalType (Id_HospitalType, HospitalType)

Hospitals (Id_Hospital, Name, Address, PostaCode, AutCommunity, Province, City, Telephone, Type, Organization)

DoctorSpecialty (Id_DoctorSpecialty, DoctorSpecialty)

Doctors (Id_Medico, Id_Hospital, Name, Specialty, PhoneNumber, Email)

The code that creates these tables in the database is the following:

```
USE [VirtualAssistantData(English)];

CREATE TABLE [HospitalType] (
      [Id_HospitalType] smallint NOT NULL,
      [HospitalType] NCHAR(35) NOT NULL
      CONSTRAINT hospital_type_pk PRIMARY KEY ([Id_HospitalType],
[HospitalType])
);

CREATE TABLE [Hospitals] (
      [Id_Hospital] int NOT NULL,
      [Name] NVARCHAR(255) NOT NULL,
      [Address] NVARCHAR(255) NOT NULL,
      [PostaCode] int,
      [AutCommunity] NVARCHAR(70),
      [Province] NVARCHAR(70),
      [City] NVARCHAR(70),
      [Telephone] int,
      [Type] smallint REFERENCES HospitalType(Id_HospitalType),
      [Organization] NVARCHAR(70)
      CONSTRAINT hospitals_pk PRIMARY KEY ([Id_Hospital])
);

CREATE TABLE [DoctorSpecialty] (
      [Id_DoctorSpecialty] smallint NOT NULL,
      [DoctorSpecialty] NCHAR(25) NOT NULL
      CONSTRAINT doctor_specialty_pk PRIMARY KEY
([Id_DoctorSpecialty],[DoctorSpecialty])
);

CREATE TABLE [Doctors] (
      [Id_Medico] int NOT NULL,
      [Id_Hospital] int NOT NULL REFERENCES Hospitals([Id_Hospital]),
      [Name] NVARCHAR(255) NOT NULL,
      [Specialty] smallint REFERENCES
DoctorSpecialty([Id_DoctorSpecialty]),
      [PhoneNumber] int,
      [Email] smallint NVARCHAR(50)
      CONSTRAINT doctors_pk PRIMARY KEY ([Id_ Medico])
);
```

4.5. New Tables Population

The created tables have to be populated with data, as RQ_9, RQ_10, RQ_11 and RQ_12 indicate. For the hospitals table the information found on the Spanish Ministry of Health, Social Services and Equality website has been used.[x] Information about all the Spanish hospitals is made public in MDB, PDF and Excel formats each year on December on the official web page of the ministry.

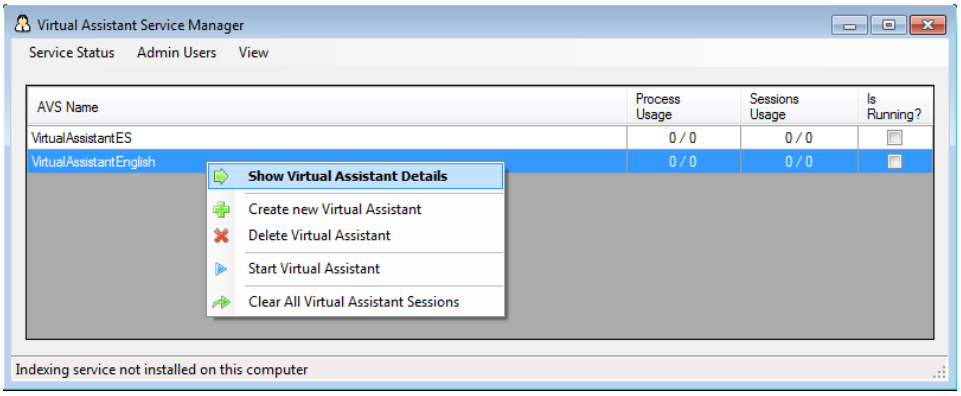
After formatting the downloaded Excel file (putting the dimension values not in separate sheets referenced to with an ID, but the values themselves, and deleting the unnecessary columns) the data is imported by right-clicking the target database (*VirtualAssistantData(English)*) and choosing *'Import Data'* from the *'Tasks'* menu. The next steps are simple: Excel must be selected as the source, and afterwards the file must be found in the file explorer. After choosing the sheet to import and accepting the changes the data will be imported in a new table, named as the sheet from which it was transferred from. After

migrating the data residing in this new table to the desired table the population has been successful. This is to be done with all the tables. Since there are dependencies, the tables must be populated in a specific order: *HospitalType* has to be populated before *Hospitals*; *DoctorSpecialty* must be populated before *Doctors*; and *Hospitals* must be populated before *Doctors*.

As for the *Doctors* table, a manual search on the Internet had to be done in order to obtain a list of doctors. Since they were saved in an Excel file, the same procedure as for Hospitals had to be done in order to import them to the respective database. A table containing the doctors' names along with all the other useful information about them is provided in Appendix A.

4.6. Virtual Assistant Service Creation

The *Virtual Assistant Service Manager*, a tool created by the KR group, is used for initiating or stopping the Windows service that makes the web interface along with all the Virtual Assistants' functionality accessible and available to the end-user. This is the tool's GUI:



It shows information about the available Virtual Assistants (two in the picture), the processes and sessions used, and whether it is running or not. By right-clicking any of them the shown menu pops up, letting the user view and modify their details, initializing it, ending the sessions, deleting it or creating a new one. Since only *VirtualAssistantES* existed in the beginning, *VirtualAssistantEnglish* had to be created. This is done by specifying a few parameters (the rest are set to default):

- The server to which it has to connect (*localhost\sqlexpress*) and the user and password for accessing it.
- The database to which it has to connect (*VirtualAssistant(English)*).
- The language it uses (English).
- The dictionary used for correcting users' typos.

4.7. Clarifier Rules Creation

Clarifier or disambiguation rules are used when each part of a phrase pattern is tried to be analyzed. Since one word can have many grammatical categories, but only one of them applying at the same time depending on the context, the VA must have rules for determining to what grammatical category a word belongs. For example, *long* can be the base of a phrasal verb or it can be an adjective, a noun or an adverb. The VA must know how to categorize a word so that it can recognize a phrase pattern (where the exact grammatical category is

specified).

The clarifier rules are best created when testing the created Phrase Patterns. The process is the following: (1) a Phrase Pattern is created using the EDR; (2) the Phrase Pattern is tested using the Indexer Tester; (3) if the Phrase Pattern is not recognized, a clarifier rule must be created or modified. Step 3 is omitted if the Phrase Pattern is recognized, and is to be repeated as many times as the test fails.

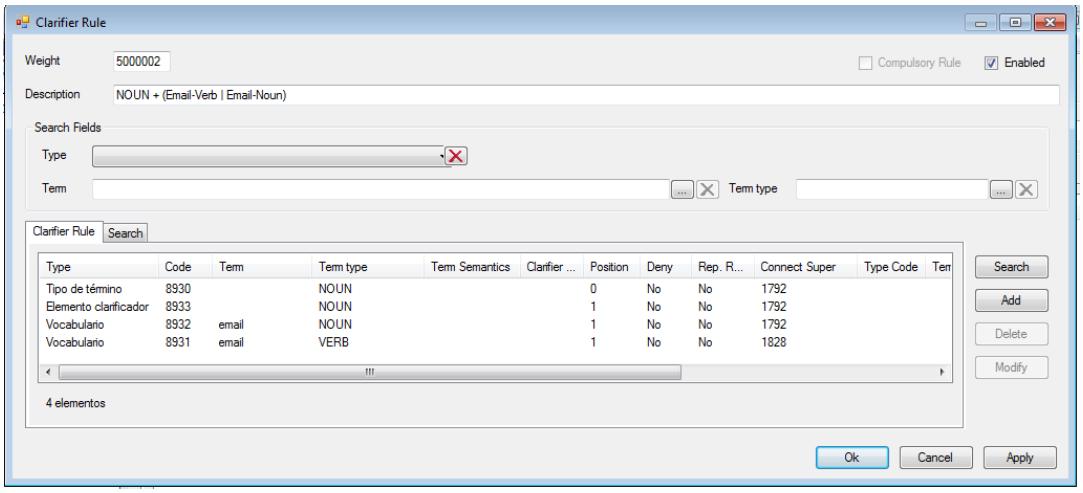
Since the *VirtualAssistant(English)* database was ready for graph recognition, there were many clarifier rules missing. The created clarifier rules are the following:

- When there is a noun followed by the word *address*, *address* will be considered to be a noun, not a verb.
- When there is a noun followed by the word *phone*, *phone* will be considered to be a noun, not a verb.
- When there is a noun followed by the word *email*, *email* will be considered to be a noun, not a verb.
- *What*, when followed by a modal verb, will be considered to be an utterance determiner, not a relative pronoun.
- When a pronoun is preceded by a phrasal verb base, the following element will be taken as a phrasal verb article, not as a preposition.

The process to create a phrasal verb is the following:

1. The Enterprise Domain Reuser tool is run.
2. From the dropdown menu under '*Ontology'*, the '*Clarifier Rules*' option is selected.
3. From the dropdown menu under '*File*', '*Add Clarifier Rule*' option is selected. In the new window a weight and a description of the rule are to be specified.
4. After saving the rule and exiting the window, the created rule must be accessed again, with more options available this time.

After this, each word must be manually inserted using the '*Add*' button. A grammatical category or a determinate word can be chosen. Its position in the disambiguation group of words must be introduced, and it is necessary to specify whether it is just a part of the disambiguation group, it is the ambiguous word, or it is a clarifier word. For example, in order to create de rule 'When there is a noun followed by the word *email*, *email* will be considered to be a noun, not a verb' the following words must be added:



A total of four words have been added: a generic noun at position 0, *email* as a verb at position 1, *email* as a noun at position 1, and the clarifier element at position 1: a noun. This means that, whenever the word *email* is preceded by a noun, it must be taken as a noun, not as a verb. This clarifier rule is used in the Phrase Pattern 'What's <hospital name>'s address?'. If address is taken as a verb, it won't be considered the pivot of the phrase.

4.8. Phrase Patterns Creation

Phrase Patterns are the core of an ontology, and therefore of the Semantic Virtual Assistant. Phrase patterns allow the VA to translate user questions into SQL queries taking into account not just independent words (as a traditional search engine would do), but acknowledging the meaning of each word in relation to the other phrase elements.

As listed in RQ_15, the following Phrase Patterns must be created:

- What is <hospital name>'s location?
- Where is <hospital name>?
- What's <hospital name>'s address?
- What oncology doctors are there at <hospital name>?
- What's <hospital name>'s phone number?
- What's <hospital name>'s address?
- What's <doctor name>'s phone number?
- What's <doctor name>'s email address?
- How can I contact <doctor name>?
- Who is <doctor name>?
- What can you tell me about <doctor name>?
- What do you know about <doctor name>?
- What hospitals in <location> have an oncology department?
- What oncology centers are there in <location>?
- Where can I find a <hospital type> hospital?
- Can you list me <number> oncology centers?

Phrase Patterns, as it is the case of Clarifier Rules, are created using the EDR tool. It is accessed from the main view of the tool, clicking on '*Ontology*' and selecting '*Phrase Patterns*'. When

adding a new Phrase Pattern the weight must be specified (the greater the weight, the earlier it is evaluated; the search for Phrase Patterns is discontinued when one has been found), as well as a description (optional) and the RSHP Pattern Type (How, When, What, etc. questions). In order to add elements to the Phrase Pattern, it must be saved by clicking 'Ok' and then it must be opened again.

The adding of words is better done in two phases:

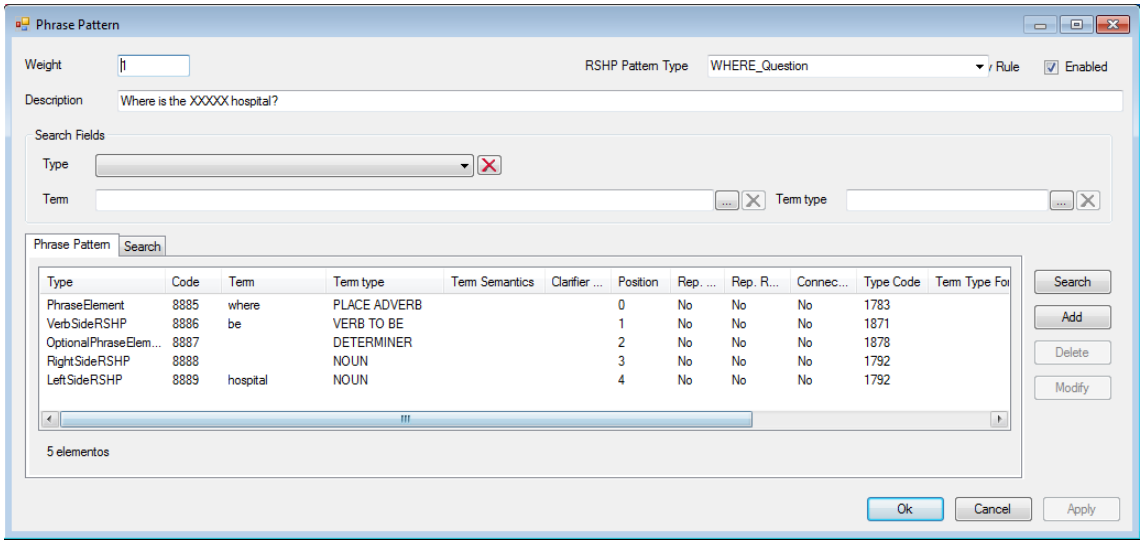
First, all the elements are added, specifying only the term or type of term and its position. All the other options are left blank. For example, for the question 'Where is the <hospital name> hospital?', the following elements would be added:

- **Where**, as a term, and 'place adverb' as a term type. Position 0.
- **Be**, as a term, and 'verb to be' as a term type. Position 1.
- A 'determiner' as a term type (no term; it stands for '**the'**). Position 2.
- A 'noun' as a term type (no term; it stands for the **hospital name**). Position 3.
- **Hospital**, as a term, and 'noun' as a term type. Position 4.

Secondly, after saving the changes, each element must be modified, adding its 'Type', RSHP code (if applicable), and concept order (if applicable). To the previous created elements it would be added:

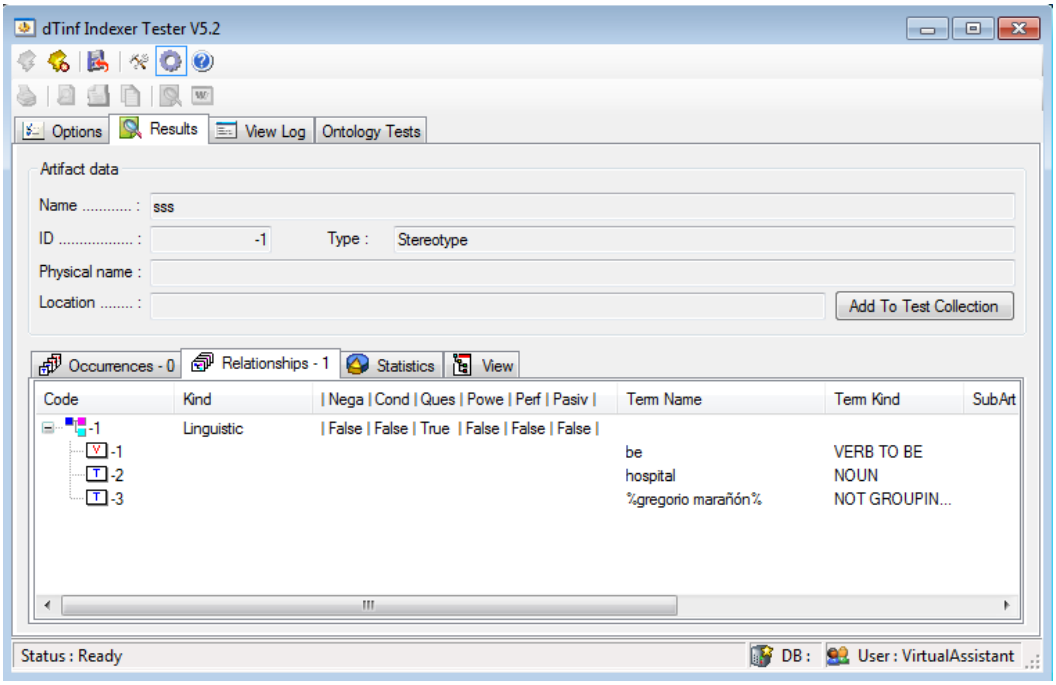
- **Where** is a 'PhraseElement' type.
- **Be** is a 'VerbSideRSHP' type. It also has the 'RSHP Code' 1.
- The determiner is an 'OptionalPhraseElement' type.
- The noun is of type 'RightSideRSHP'. It has 'RSHP Code' 1 and 'Concept Order' 2.
- **Hospital** is of type 'LeftSideRSHP'. It has 'RSHP Code' 1 and 'Concept Order' 1.

The final result can be seen in the following picture:



The next step is testing if the Phrase Pattern will be recognized by the SVA. This is done with the Indexer Tester, another tool that is part of the VA's ecosystem. After opening it, the credentials must be introduced, and the Indexer chosen when the program asks for it. In the

Indexer Configuration ( ) must be clicked and 'Paragraph' selected as the Index source. Then the Index icon ( ) must be clicked. In the Options tab, an artifact has to be selected (any of the available), and then the question can be introduced in the 'Paragraph to index' section. The 'Index' button has to be selected, and in the Results tab the outcome can be observed:



The relationship has been recognized: the Phrase Pattern has been successfully built, and the SVA is ready to recognize questions of type 'Where is the <hospital name> hospital?'

This process must be repeated for each question that is to be turned into a Phrase Pattern.

4.9. SQL Statements Creation

Although the SVA can recognize questions that match the created Phrase Patterns, it also must have an SQL query associated in order for it to be able to answer. For the created phrase pattern the following SQL query has to be created:

```
select 'The address for that hospital is ' + address from hospitals
where name like '%' + ? + '%'
```

The question mark represents the right side RSHP, which is the hospital name.

4.10. Structured Questions rows Creation

The last step to be done for the SVA to be able to answer questions is adding the required rows in the *Structured_Questions* table from the *VirtualAssistant(English)* database. The following fields must be added:

- ID, which identifies the structured question.
- Pivot. The left side RSHP as created in the Phrase Patterns. Continuing with the previous question, the pivot is *hospital.*

- QuestionType: the chosen question type identifier, 70 for *Where*
- NoResultAnswer: the desired result to be retrieved if it doesn't find anything in the database.
- Question, in our case, 'Where is the xxxx hospital'.
- VerbSemantic: the VerbSideRSHP, the verb *to be*, with the ID 1474.
- NumVariables: the question's number of variables, 1 in this case.
- SQLStatement: here the SQL statement previously created must be inserted.
- DatabaseServer, DatabaseName, DatabaseLogin and DatabaseType, which are the credentials for connecting to the database.

The rest of the fields should not be modified but let by default.

4.11. Testing

Having completed all the previous steps, the Virtual Assistant is now ready to be tested. First the service must be initialized using the Virtual Assistant Service Manager tool. Afterwards the local IP address and the service's port must be concatenated in the URL of a web browser (the port is specified in the IIS utility and the IP address can be substituted with *localhost*). A new window is opened when the VA's photo is clicked. Here the question can be introduced, and the result can be seen in the following screenshot:

The SVA is then able to understand questions and provide answers to the user.

5. Conclusions and Further Improvements

The Semantic Virtual Assistant can be a helpful tool if used as a directory for hospitals in general and oncology-related hospitals and doctors in particular. Its friendly interface is surely more welcoming than complex webpages filled with information; the elderly may find web pages daunting, but a kind avatar really can highly simplify things for them. The most valued feature, however, is that users can make questions using human-like speech and receive meaningful answers: that is the semantic part of the Semantic Virtual Assistant.

This doesn't mean that it is perfect; a few improvements can be listed for future versions:

- The ability to recognize even more questions. That is, adding more Phrase Patterns along with the rest of work to be done.
- Having more doctors. The ideal scenario would be to obtain a database with all the (oncology) doctors from all the Spanish hospitals.
- To include more health information. For example, it could have definitions for various diseases (or types of cancer), advices to follow if a person has any type of cancer, and suggestions for living a healthy life.

The SVA ecosystem, now considering the technical (back-end) part, can be highly improved; but this has already been done in newer versions by the KR group. For example, a much easier, visual and intuitive way of creating Phrase Patterns has been implemented. The Indexer Tester has also been improved. But the best part is that they are all part of the same tool, unlike the version used for this project.

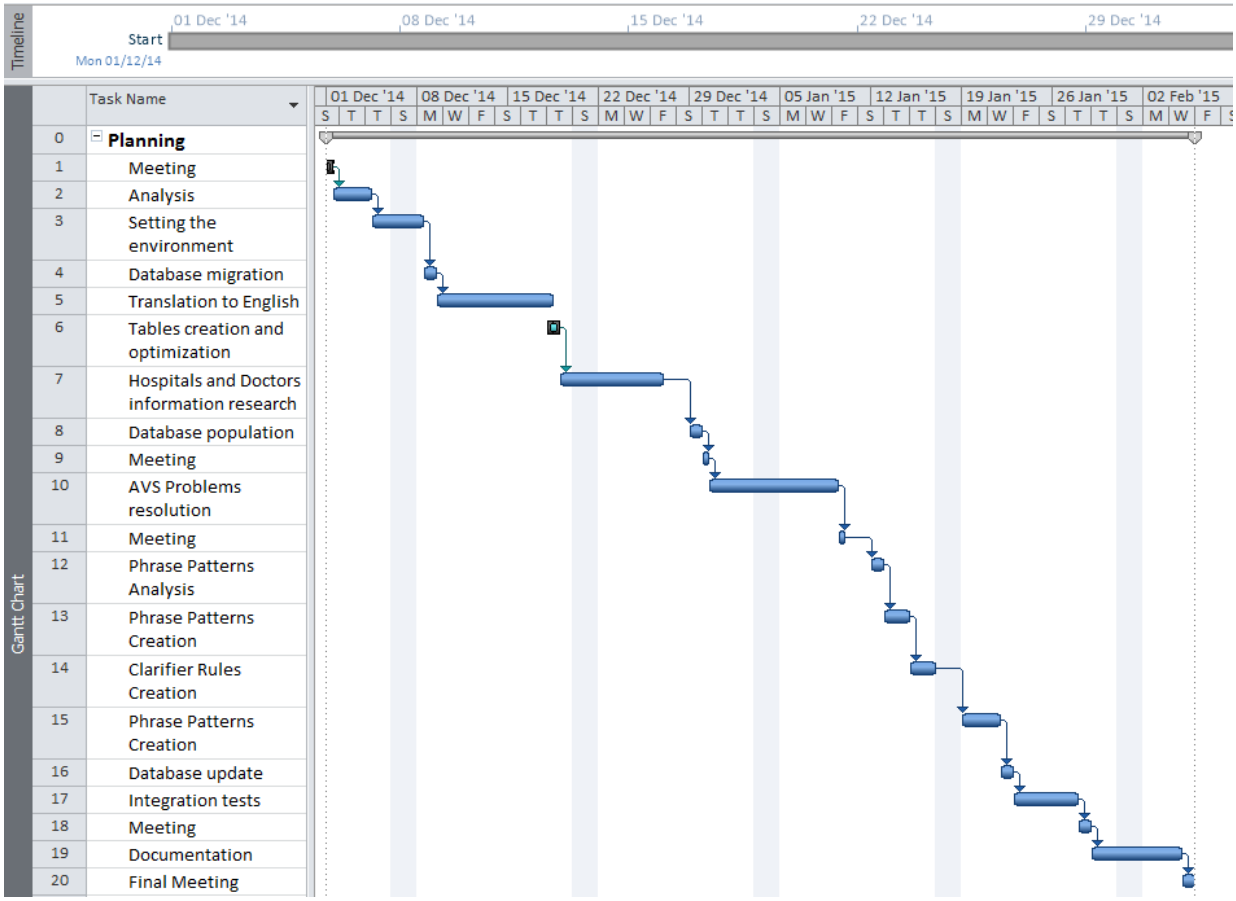
6. Planning and Budget

The project has been developed over a total of 45 working days. If the project is reckoned to have started on the first of December, 2014, its finish date has been on the fifth of February, 2015. The project's tasks and their duration and starting and finish date can be seen in the following table:

Task Name	Duration	Start	Finish
Planning	**45 days**	**Mon 01/12/14**	**Thu 05/02/15**
Meeting	0,5 days	Mon 01/12/14	Mon 01/12/14
Analysis	3 days	Mon 01/12/14	Thu 04/12/14
Setting the environment	2 days	Thu 04/12/14	Mon 08/12/14
Database migration	1 day	Mon 08/12/14	Tue 09/12/14
Translation to English	7 days	Tue 09/12/14	Thu 18/12/14
Tables creation and optimization	1 day	Thu 18/12/14	Thu 18/12/14
Hospitals and Doctors information research	5 days	Fri 19/12/14	Fri 26/12/14
Database population	1 day	Mon 29/12/14	Mon 29/12/14
Meeting	0,5 days	Tue 30/12/14	Tue 30/12/14
AVS Problems resolution	5 days	Tue 30/12/14	Fri 09/01/15
Meeting	0,5 days	Fri 09/01/15	Fri 09/01/15
Phrase Patterns Analysis	1 day	Mon 12/01/15	Mon 12/01/15
Phrase Patterns Creation	2 days	Tue 13/01/15	Wed 14/01/15
Clarifier Rules Creation	2 days	Thu 15/01/15	Fri 16/01/15
Phrase Patterns Creation	3 days	Mon 19/01/15	Wed 21/01/15
Database update	1 day	Thu 22/01/15	Thu 22/01/15
Integration tests	3 days	Fri 23/01/15	Tue 27/01/15
Meeting	1 day	Wed 28/01/15	Wed 28/01/15
Documentation	5 days	Thu 29/01/15	Wed 04/02/15
Final Meeting	1 day	Thu 05/02/15	Thu 05/02/15

Weekends, of course, are not taken into account, and the 25th and 31st of December, the 1st and 6th of January are set as holidays, and therefore not taken into account either.

The tasks along with their visual representation over time can be observed in the following Gantt chart illustration:



As for the budget, it has been computed as if two people have been working on the project: a Programmer and an Analyst. The Programmer has been working from the first to the last day (45 days), while the Analyst participated only in the management of the project (5 days), in meetings (3.5 days), analysis (4 days) and documentation (5 days).

The Programmer is paid (or better, costs) 20 € per hour, while the Analyst costs 30 € per hour. The total cost is reflected in the next table:

Person	Days	Hours	Cost per hour	Total Cost
Analyst	17.5	140	30	4200
Programmer	45	360	20	7200
Analyst + Programmer	62.5	500	-	**11400**

Therefore, the total cost of the project has amounted to **11,400 €.**

7. References and Bibliography

[i] It must be noted that there are around 161 synonyms for this concept. Van, L. E. (n.d.). *161 Humanlike conversational AI synonyms*. Retrieved from https://www.chatbots.org/synonyms/
[ii] Moor H. J. (2001). *The Status and Future of the Turing Test*. Netherlands: Kluwer Academic Publisher
[iii] Epstein R., Roberts G., & Beber G. (2009*). Parsing the Turing Test: Philosophical and Methodological Issues in the Quest for the Thinking Computer* (pp. 185). Netherlands: Springer
[iv] Loebner H. (n.d.). *Home Page of The Loebner Prize in Artificial Intelligence "The First Turing Test".* Retrieved from http://www.loebner.net/Prizef/loebner-prize.html
[v] Majid M. R. (2014). *The society for the study of Artificial Intelligence and Simulation of Behaviour. Loebner prize 2014*. Retrieved from http://www.aisb.org.uk/events/loebner-prize
[vi] Javanmardian M., & Choudhury J. S. (2014). *2014 Priorities for the Healthcare Industry*. Retrieved from http://www.forbes.com/sites/boozandcompany/2014/03/31/2014-priorities-for-the-healthcare-industry/
[vii] Akay M., & Marsh A. (2001). *Information Technologies in Medicine, Volume I: Medical Simulation and Education.* USA, Canada: John Wiley & Sons, Inc.
[viii] Max Visual Basic. *History of Visual Basic.* (2010). Retrieved from http://www.max-visual-basic.com/history-of-visual-basic.html
[ix] Waymire R., & Sawtell R. (1999). *Sams Teach Yourself Microsoft SQL Server 7 in 21 Days.* USA: Sams Publishing
[x] Gobierno de España. Ministerio de Sanidad, Servicios Sociales e Igualdad. Catálogo de Centros de Atención Primaria del SNS y Catálogo Nacional de Hospitales. (http://www.msssi.gob.es/ciudadanos/prestaciones/centrosServiciosSNS/hospitales/home.htm)